

## Contents

<b>1 Routine/Function Prologues</b>	<b>2</b>
1.0.1 zterp.F90 (Source File: zterp.F90) . . . . .	2
1.0.2 coszenith (Source File: zterp.F90) . . . . .	6
1.0.3 localtime (Source File: zterp.F90) . . . . .	8

## 1 Routine/Function Prologues

### 1.0.1 zterp.F90 (Source File: zterp.F90)

This subroutine is based, in part, on modified subroutines from Jean C. Morrill of the GSWP project. The program temporally interpolates time average or instantaneous data to that needed by the model at the current timestep. It does this by combining a linear interpolation approach with a solar zenith angle approach in a fashion suitable for use with data such as short wave radiation values. It cannot be used with input data points which are more than 24 hours apart. The program outputs two weights which can then be applied to the original data to arrive at the interpolated data. If IFLAG=0, then WEIGHT1 is the weight which should be applied to the time averaged data (from the time period which the model is currently in) to arrive at the interpolated value and weight 2 is not used at all. If IFLAG=1, then WEIGHT1 should be applied to the original instantaneous data located just prior to the model time step, and WEIGHT2 should be applied to the original instantaneous data located just after the model time step.

i.e. (IF IFLAG=0) interpolated data = (WEIGHT1 \* time averaged data from time period that model is currently in)

i.e. (IF IFLAG=1) interp. data = (WEIGHT1\*past data)+(WEIGHT2\*future data)

#### REVISION HISTORY:

```

10/2/98 Brian Cosgrove
6/28/00 Brian Cosgrove; changed code so that it uses LDAS%UDEF and
not a hard-wired undefined value of -999.999. Also changed
call to ZTERP subroutine so that LDAS and GRID mod files
are brought in and can be accessed in this subroutine
2/27/01 Brian Cosgrove; Added czmodel into call for ZTERP subroutine
10/7/01 Urszula Jambor; Added conditional to prevent stop in code when
AGRMET data are available, but both endtime cos(zen.) are small.

```

#### INTERFACE:

```

subroutine zterp (iflag,lat,lon,btime,etime, &
mbtime,julianb,weight1,weight2,czbegdata, &
czenddata,czmodel,lis)

```

#### USES:

```

use lis_module      ! LDAS non-model-specific 1-D variables
implicit none
!INPUT PARAMETERS
type (lisdec) :: lis
integer         :: iflag    !Flag specifying if input data is
                           !time averaged or not
real            :: lat,lon !Latitude and longitudes of current
                           !data point
real            :: mbtime   !Time of current time step Expects
                           !GMT time in hour fractions
integer         :: julianb  !Julian day upon which BTIME falls
real            :: interval

```

```

real          :: btime   !beginning time of orig. avg. data (IFLAG=0) or
                  !time of original instantaneous data point which
                  !is located at or just prior to the current model
                  !time step (IFLAG=1).  Expects GMT time in hour
                  !fractions. i.e., 6:30 Z would be 6.5.
real          :: etime    !Ending time of orig. avg. data (IFLAG=0) or
                  !time of original instantaneous data point which
                  !is located at or just after the current model
                  !time step (IFLAG=1).  Expects GMT time in hour
                  !fractions. i.e., 6:30 Z would be 6.5.
real          :: weight1  !weight applied to original time averaged
                  !data (IFLAG=0) or
                  !weight applied to orig instantaneous data point
                  !located just prior to the current model time step
real          :: weight2  !weight applied to orig instantaneous
                  !data point located just after the current model
                  !time step (IFLAG=1)
                  !If IFLAG=0, then this weight is meaningless and
                  !should not be used

```

## CONTENTS:

```

!-----
! This section contains hardwired data that will be supplied by main program.
! These values were chosen arbitrarily and exist simply to check the
! functioning of the program.
!
! Initialize variables
!-----
i=1
totangle=0
weighte=lis%d%udef
weightb=lis%d%udef
weight1=lis%d%udef
weight2=lis%d%udef
czbegdata=lis%d%udef
czenddata=lis%d%udef
czmodel=lis%d%udef
gmt=btime
juliane=julianb
julianmb=julianb
juliantemp=julianb

if (mbtime.lt.btime) julianmb=julianmb+1
if (etime.le.btime) juliane=juliane+1
!-----
! First case, IFLAG=0 (Time average input, instantaneous output)
! Compute time interval, here arbitrarily divided into 36 parts
!-----
```

```
if (iflag.eq.0) then
    call localtime (mbtime,lon,lhour,zone)
    call coszenith(lon,lat,lhour,zone,julianmb,czmodel)
    if (czmodel.eq.0) then
        weight1=0
        goto 818
    endif

    if (etime.gt.btime) then
        interval = ((etime-btime)/36.0)
    elseif (etime.lt.btime) then
        interval = (((24-btime)+(etime))/36.0)
    else
        interval=24.0/36.0
    endif
!-----
!      Compute cosine of zenith angle for each time interval
!-----
do while (i.le.37)
    if ((gmt+interval).lt.24) then
        gmt=gmt+interval
    else
        gmt=(interval-(24-gmt))
        juliantemp=juliantemp+1
    endif
    call localtime (gmt,lon,lhour,zone)
    call coszenith(lon,lat,lhour,zone, &
                  juliantemp,czavgdata)
    totangle=totangle+czavgdata
    i=i+1
enddo
!-----
!      Compute average cosine of zenith angle and also
!      weight which will be applied to original data (WEIGHT1
!-----
avgangle=(totangle/37.0)
if (avgangle.eq.0) then
    weight1=0
else
    weight1=(czmodel/avgangle)
endif
endif
!-----
!      Second case: IFLAG=1 (instantaneous input and output)
!-----
if (iflag.eq.1) then
!-----
!      Compute local times and cosine (zenith angle)
```

```

!-----
      call localtime (btime,lon,lctime,zonebtime)
      if (lctime.gt.btime) julianb=julianb-1
      call localtime (etime,lon,letime,zoneetime)
      if (letime.gt.etime) juliane=juliane-1
      call localtime (mbtime,lon,lmbtime,zonembtime)
      if (lmbtime.gt.mbtme) julianmb=julianmb-1
      call coszenith (lon,lat,lctime,zonebtime, &
                      julianb,czbegdata)
      call coszenith (lon,lat,letime,zoneetime, &
                      juliane,czenddata)
      call coszenith (lon,lat,lmbtime,zonembtime, &
                      julianmb,czmodel)
!-----
!      Decision tree to deal with contingencies
!      If COS(zenith angle at current model time =0, weight =0
!      If COS(zenith angle =0 at beg. and end times, PROBLEM, STOP
!      Otherwise use beginning and ending data to calculate weight
!-----
      if (czmodel.le.0.01) then
          weight1=0
          weight2=0
      else
          if ((czbegdata.gt.0.01).or.(czenddata.gt.0.01)) then
              if (czbegdata.le.0.01) then
                  weight1=0
                  weight2=(czmodel/czenddata)
              endif
              if (czenddata.le.0.01) then
                  weight1=(czmodel/czbegdata)
                  weight2=0
              endif
              if((czenddata.gt.0.01).and. &
                 (czbegdata.gt.0.01))then

                  if (btime.le.mbtme) then
                      diffbm=mbtime-btime
                  else
                      diffbm=24-btime+mbtime
                  endif

                  if (etime.ge.mbtme) then
                      diffem=etime-mbtme
                  else
                      diffem=24-mbtme+etime
                  endif

                  if (etime.gt.btime) then

```

```

        diffeb=etime-btime
        elseif (etime.eq.btime) then

            diffeb=24
        else
            diffeb=24-btime+etime
        endif
        weighte=(diffbm/diffeb)
        weightb=(diffem/diffeb)

        weight1=((czmodel/czbegdata)*weightb)
        weight2=((czmodel/czenddata)*weighte)

    endif
else
    print*, 'no data to interpolate to/from'
    print*, 'beginning and ending data both = 0'
    print*, 'stopping!!!'

    call endrun
endif
endif
endif
endif
818 continue
return

```

---

### 1.0.2 coszenith (Source File: zterp.F90)

- 1) Day angle (GAMMA)
- 2) Solar DEClination
- 3) Equation of time
- 4) Local apparent time
- 5) Hour angle
- 6) Cosine of zenith angle

All equations come from "An Introduction to Solar Radiation" By Muhammad Iqbal, 1983.

**INTERFACE:**

```
subroutine coszenith (lon,latd,lhour,zone,julian,czenith)
```

```
implicit none
```

**ARGUMENTS:**

```

integer :: zone          ! time zone (1-24) gmt=12
integer :: julian         ! julian day
real :: czenith           ! cosine of zenith angle (radians)

```

```

real :: dec                      ! solar declination (radians)
real :: et                        ! equation of time (minutes)
real :: gamma                     ! day angle (radians)
real :: latime                    ! local apparent time
real :: lcorr                      ! longitudinal correction
real :: lhour                     ! local standard time
real :: lon                        ! local longitude (deg)
real :: llat                        ! local latitude in radians
real :: latd                        ! local latitude in degrees
real :: ls                          ! standard longitude (deg)
real :: omegad                     ! omega in degrees
real :: omega                      ! omega in radians
real :: pi                         ! universal constant pi [-]
real :: zenith                     ! zenith angle(radians)

```

## CONTENTS:

```

!-----
! Neither ZENITH nor ZEND are necessary for this program.
! I originally used them as checks, and left them here in
! case anyone else had a use for them.
!
! 1) Day angle GAMMA (radians) page 3
!-----
pi= 3.141592                  ! universal constant pi
gamma=2*pi*(julian-1)/365.
!-----
! 2) Solar declination (assumed constant for a 24 hour period) page 7
! in radians
!
dec=(0.006918-0.399912*cos(gamma)+0.070257*sin(gamma) &
     -0.006758*cos(2*gamma)+0.000907*sin(2*gamma) &
     -0.002697*cos(3*gamma)+0.00148*sin(3*gamma))
!
! maximum error 0.0006 rad (<3'), leads to error of less than 1/2 degree
! in ZENITH angle
!
! 3) Equation of time page 11
!-----
et=(0.000075+0.001868*cos(gamma)-0.032077*sin(gamma) &
     -0.014615*cos(2*gamma)-0.04089*sin(2*gamma))*229.18
!
! 4) Local apparent time page 13
!
! LS      standard longitude (nearest 15 degree meridian)
! LON     local longitude
! LHOUR   local standard time
! LATIME  local apparent time
! LCORR   longitudunal correction (minutes)

```

```

!-----
ls=((zone-1)*15)-180.
lcorr=4.*(ls-lon)*(-1)
latime=lhour+lcorr/60.+et/60.
if (latime.lt.0.) latime=latime+24
if (latime.gt.24.) latime=latime-24
!-----
!      5) Hour angle OMEGA  page 15
!
!      hour angle is zero at noon, positive in the morning
!      It ranges from 180 to -180
!
!      OMEGAD is OMEGA in degrees, OMEGA is in radians
!-----
OMEGAD=(LATTime-12.)*(-15.)
OMEGA=OMEGAD*PI/180.
!-----
!      6) Zenith angle page 15
!
!      CZENITH cosine of zenith angle (radians)
!      LATD=local latitude in degrees
!      LLAT=local latitude in radians
!-----
llat=latd*pi/180.
czenith=sin(dec)*sin(llat)+cos(dec)*cos(llat)*cos(omega)
czenith=amax1(0.,czenith)
zenith=asin(czenith)
return

```

---

### 1.0.3 localtime (Source File: zterp.F90)

Calculates the local time based on GMT

INTERFACE:

```
subroutine localtime (gmt,lon,lhour,zone)
```

ARGUMENTS:

real:: gmt	! GMT time (0-23)
real:: lon	! longitude in degrees
real:: change	! the change in number of hours between
real:: lhour	! local hour (0-23) 0= midnight, 23= 11:00 p.m.
integer:: i	! working integer
integer:: zone	! time zone (1-24)

CONTENTS:

```
!-----
! Determine into which time ZONE (15 degree interval) the
! longitude falls.
!-----
do i=1,25
  if (lon.lt.(-187.5+(15*i))) then
    zone=i
    if (zone.eq.25) zone=1
    go to 60
  end if
end do
!-----
! Calculate change (in number of hours) from GMT time to
! local hour. Change will be negative for zones < 13 and
! positive for zones > 13.
! There is also a correction for LHOUR < 0 and LHOUR > 23
! to LHOUR between 0 and 23.
!-----
60 if (zone.lt.13) then
  change=zone-13
  lhour=gmt+change
elseif (zone.eq.13) then
  lhour=gmt
else
  change=zone-13
  lhour=gmt+change
end if
if (lhour.lt.0) lhour=lhour+24
if (lhour.gt.23) lhour=lhour-24
return
```